

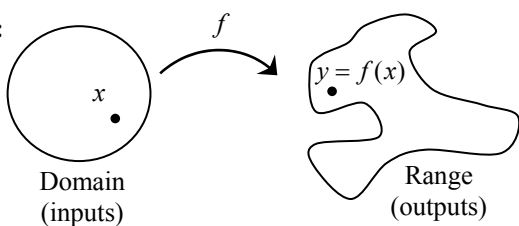
### Functions Framework

**Terminology:**

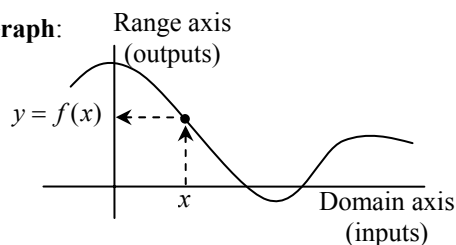
- The *domain*,  $X$ , is the set of inputs.
- The *range*,  $Y$ , is the set of possible outputs.
- In this class, our domain and range will both usually be the *real numbers*,  $\mathbf{R}$ .
- Each input must produce a *unique output*.
- Different inputs may produce the same output. For example  $f(x) = x^2$  has  $f(-2) = f(2) = 4$ , so the output 4 is produced by inputs of  $-2$  and  $2$ .
- If each input produces a different output, we say the function is *one-to-one*. For example,  $f(x) = 3x - 5$  produces a different output for every input.
- If  $f(x_1) < f(x_2)$  whenever  $x_1 < x_2$  on some interval, we say that  $f$  is *increasing* on that interval.
- If  $f(x_1) > f(x_2)$  whenever  $x_1 < x_2$  on some interval, we say that  $f$  is *decreasing* on that interval.

**Notation:**  $f(x) = y$  means that  $f$  maps the input  $x$  to the output  $y$ .

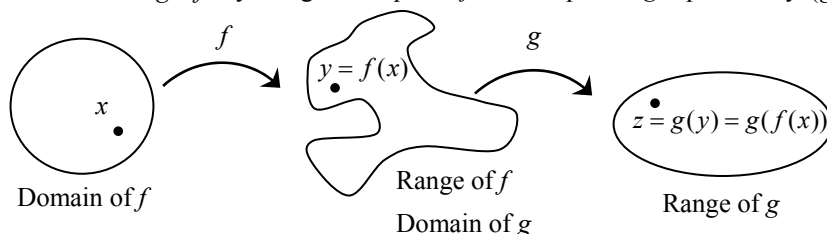
**Diagram:**



**Graph:**

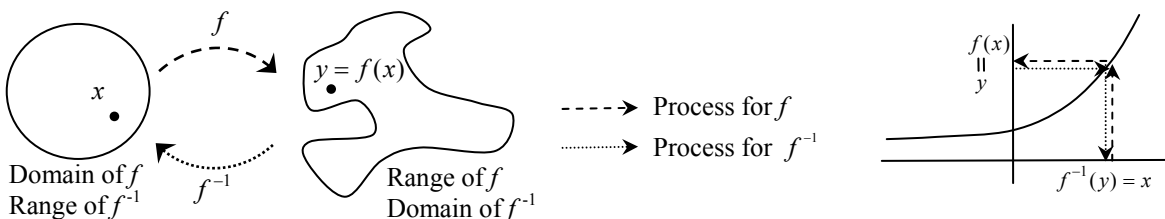


**Composition:** If the range of one function,  $f$ , is within the domain of a second function  $g$  then we can compose them to get a new function  $g \circ f$  by using the output of  $f$  as the input for  $g$ . Specifically  $(g \circ f)(x) = g(f(x))$ .



**Inverse:** If  $f : X \rightarrow Y$  is one-to-one, then there is an inverse function  $f^{-1} : Y \rightarrow X$  that undoes the action of  $X$ .

That is, if  $f(x) = y$  then  $f^{-1}(y) = x$



**Warning:** The inverse notation looks a lot like an exponential power of  $-1$ . However,  $f^{-1}(x)$  and  $f(x)^{-1}$  are two very different things!

$$f^{-1}(x) \text{ means function inverse}$$

$$f(x)^{-1} \text{ means reciprocal or } \frac{1}{f(x)}$$

**Note:** If you can explain why  $(g \circ f)^{-1} = f^{-1} \circ g^{-1}$  (and not  $g^{-1} \circ f^{-1}$ ) then you probably understand the ideas of composition and inverse pretty well.

**Functions in MATLAB:** Often you will want to define a function in MATLAB so you can call it repeatedly without having to type in the whole formula each time. Then you can also change the formula once, and every time it is called in other places in the program will use the updated formula! There are two main ways to work with functions:

**Symbolic Expressions:** First define your input variable as a symbolic variable. If it is  $x$ , you do this with the command

```
syms('x');
```

Then you can define a function in terms of  $x$  with a command like

```
f=5*exp(-x)*sin(x);
```

Now in order to evaluate the function at some point use the 'subs' command. For example if you wanted to find  $f(\pi)$ , you would type

```
subs(f,pi)
```

The advantage of using symbolic expressions is that MATLAB can operate on them algebraically. For example, if you type

```
syms('x');
simplify((x^2-4)/(x+2))
```

you will get back the result

```
ans = x-2
```

You can plot symbolic expressions with the ezplot command or by making a vector plot:

```
ezplot(f)                x=-5:0.01:5;
                        or  y=subs(f,x);
                        plot(x,y)
```

For additional information on symbolic expressions, see Chapter 8 in the manual.

**Function m-files:** Create a new m-file and make the first line

```
function f=f(x)
```

Then you can write MATLAB command just as in any m-file that will compute the value of  $f(x)$ . An advantage of this is that you can create functions that require several steps to compute. A simple example is a piecewise function.

You could create the function  $f(x) = \begin{cases} x^2 + 2x - 3, & x \leq 0 \\ 2x - 3, & x > 0 \end{cases}$  using the MATLAB M-file

```
function f=f(x)
if x <= 0
    f=x^2+2*x-3;
else
    f=2*x-3;
end
```

In addition to being able to create complex programs to compute a function, another advantage to using m-files is that you then use standard function notation. So to find  $f(2)$ , you would simply type

```
f(2)    or    y=f(2)
```

An m-file function can have multiple inputs and multiple outputs simply by altering the first line. For example, a function with  $x$  and  $y$  as inputs and  $r$  and  $\theta$  as outputs can be defined by

```
function [r,theta]=f(x, y)
```

Then to call the function, use a command such as

```
f(x,y)    or    [r,theta]=f(x,y)
```

To graph a function defined in an m-file, you can use the following vector plot method:

```
x=-5:0.01:5;
for j=1:length(x)
    y(j)=f(x(j));
end
plot(x,y)
```

In an m-file function, you cannot access any variables other than the ones provided as inputs and those defined within the m-file itself. Also, the only variable that will be saved after the m-file function has run are those defined as outputs.

For additional details on m-file functions, see the MATLAB help on "functions" and pp. 6-19 through 6-21 in the manual.